Genesis

The Complete Adventure Creation System

AMSTRAD CPC 464/664

CONTENTS

About Genesis

Consisting of three utilities for creating a professional quality Adventure game, plus a free example game, Genesis provides everything needed to design text Adventures, complete with Sound and Graphics with only a minimum of computer expertise. The games produced will run independently of Genesis when complete.

Most of the structure and mapping of your game is done using the Texture utility. With this all the game Actions are designed, commands defined and locations and objects/characters set up. At the same time, arrangements are made to activate Picture/Sound blocks at the correct time, eg. when changing locations or firing a gun. The actual pictures and sounds are created using DEPICTER, which has full editing facilities on both sound and graphics. Up to 20 separate graphic windows can be defined, and pictures plotted in different windows at different times if desired. A split-mode screen option allows the combination of say, 16-colour graphics and 80 column text on the same screen, a great boon for graphics adventures.

Sound wise, up to three sound tracks can be laid down and synchronised for each picture, with full control over envelope shaping for special effects.

Both Texture and DEPICTER produce data files on cassette, which are eventually loaded into the CLONER program to produce the finished game.

Note:

In what follows, we have enclosed things to type in $ signs, eg. RUN. The actual dollar signs are not typed. Where the ENTER key is required, this is illustrated by a pair of dollar $$.

The first steps. Cloning the example Adventure.

As an easy introduction to Genesis, we will generate a stand-alone game from the example Adventure. For this, Side 2 of the cassette is used, containing the CLONER program, plus the two data files from Texture and DEPICTER. You will also need a blank tape (C15 is OK if you save on Fast speed).

First, zero the tape counter, then run CLONER using:-
$RUN""$$
(It is always best to start with a clean machine before running a new program. Press SHIFT, CTRL, and ESC together to clear it, before typing RUN).

When the tape stops, leave it on PLAY, but press the PAUSE key while you select the save speed for your output tape. Then raise PAUSE and wait for the tape to stop again.

Normally at this point you would rewind the CLONER tape and load your blank tape. However, since the data files follow the CLONER program, rather than being on a separate tape, just remove the CLONER tape without rewinding. It is a good idea to note the reading on the tape counter at this point, as we shall need to locate the data files again in later sections of the manual.

Insert the output tape, making sure you have advanced it past the plastic leader tape. This is important - Genesis does not wait for the leader to pass through before starting to save data. Put the datacorder to RECORD, and press any key to save the first section.

Now remove the output tape (do not rewind it), and re-insert the CLONER tape (or of course your own data files if you are cloning your own Adventure). Before reading the data, CLONER will ask for the following parameters:-

1. The Initial Action Number. Type in $26$$ for the example. As explained later, Action are used in Genesis to do things such as printing messages, displaying pictures, etc. They are numbered Ø to 249, and every game must have one set up to create the initial screen pictures and messages, and to put the player into the start location. Of course, it is up to you what Action number you use, and what the Action does.

2. Pen Colour Selection

   There now follow 3 prompts asking for the inks for pens 1, 2 and 3. Pen 1 is used to print messages, pen 2 for player inputs, and pen 3 for location names. The ink selected must be sensible for the screen mode, as follows:-

   Mode Ø Inks Ø to 15 valid
   Mode 1 Inks Ø to 3 only
   Mode 2 Inks Ø or 1 only

   Of course, you would not normally use Ink Ø, as this would make the text invisible! If you choose a higher value than is valid, this can also cause the same effect.

   It is assumed that you are aware of the difference between screen modes, if not, look it up in your Amstrad User Manual. Genesis allows you to have split-mode screens (ie. top and bottom of the screen having different modes), as well as the normal single mode screen. This is ideal for mixing 8Ø column text with 4 or 16 colour graphics on the same screen.

   For the example Adventure, respond $1$$,$1$$, and $3$$ respectively to the three prompts.

Now CLONER is ready to read the data files. Press PLAY, and any key to start loading these. Finally, re-insert the output tape to save the second main section of the game. You should now have an independent game which can be loaded using $RUN""$$ (clear the computer first).

Playing the Adventure

We don't tell you how to solve the Adventure, you will have to work it out! However, there are a few commands which are common to all Genesis Adventures, in addition to those that you create yourself. These are:-

1. Save and Load

   Fairly obvious, these are used to save partly complete games, so that the player can start from where he left off.

2. Quit

   This effectively restarts from the beginning, resetting all objects, etc. to their initial status and re-activating the Initial Action.

3. CTRL A

   Holding down the CTRL key and pressing key A will cause any picture being drawn to be skipped.

4. Upper/Lower Case

   Generally speaking, Genesis ignores any Upper/Lower case letter differences (ie. SAVE is the same as save).

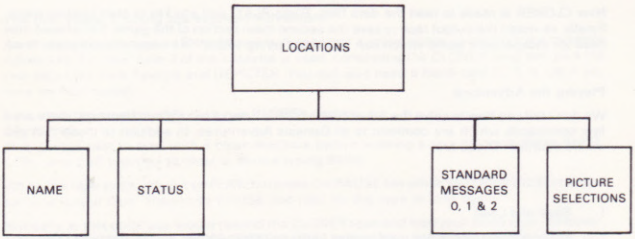

Fig. 1 LOCATIONS

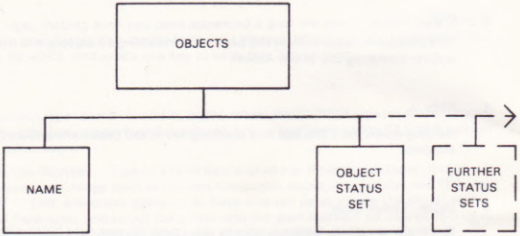

Fig. 2. OBJECTS

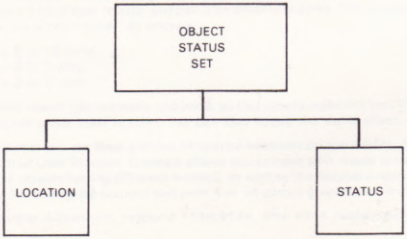

Fig. 3. OBJECT STATUS SET STRUCTURE

COMPONENT STRUCTURES IN TEXTURE

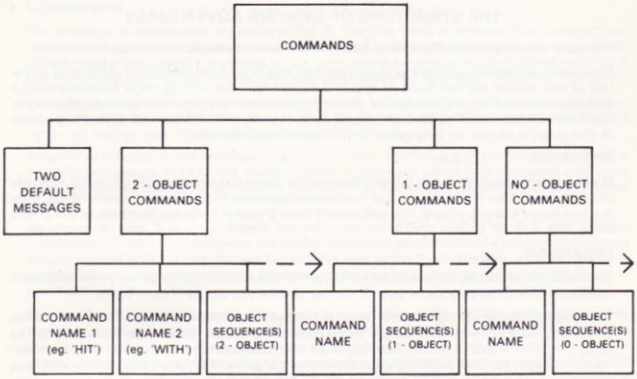

Fig. 4. STRUCTURE OF COMMANDS

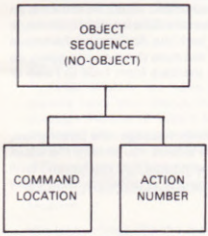

Fig. 5. 0-OBJECT SEQUENCE

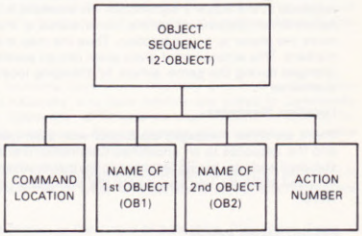

Fig. 7. 2-OBJECT STRUCTURE

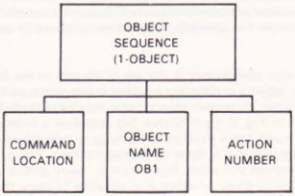

Fig. 6. 1-OBJECT STRUCTURE

THE STRUCTURE OF GENESIS ADVENTURES

We must now begin to probe a little deeper into the components of Genesis Adventures.

Most users of Genesis will have at least a general idea of what an Adventure consists of. In the rest of this section we will build on this and explain how the various major components of a Genesis adventure are linked together. In fact there are only five major components, Messages, Locations, Objects, Commands and Actions. Each of these will be expanded upon. The purpose of this stage is to give an overview - do not worry about details.

MESSAGES

Messages are simply all those things you want the player to see during the game, for example the descriptions for the locations and the visible responses to all the players commands. They are numbered starting at zero. You will need to keep a note of message numbers as you create them.

LOCATIONS

Locations are, of course, the places which the player can visit during the game. In Genesis, each location is identified by a name (which can be several words), as shown in Fig. 1.

You may already be asking how the 'map' is created, ie. what determines the directions the player can move in from each location. This is done using 'status' markers, each of which can be either 'True' or 'False'. You can have up to 24 such markers representing not only directions 'north', 'south', etc., but anything else relevant to the game. It is up to you to decide what you want to use them for. You might decide to use one to determine whether that location was 'dark' or not, for example.

Stratus markers are used in Actions - described later - in order to control what happens in response to the player's commands. For example, a 'go north' command would be linked to an Action which tested whether the 'north' status is 'true' for the current location, and if so would move the player to a new location. Thus the map is defined by both the Action and the status markers. This arrangement gives great design possibilities, for example the map can easily be changed during the game, simply by changing location status markers from True to False or vice-versa.

Location Messages

There are three messages associated with each location, the Help message, the Description, and the response to an undefined Command. The last of these allows you to vary the usual standard and very boring response to the player when he uses a command that you haven't built in. Used skillfully, you can create the impression that the computer is understanding every word!

Picture/Sound Blocks

Pictures and sounds are created separately with the DEPICTER utility, and two different blocks can be associated with each location. These are normally the ones the player would see at that location, though it is possible to put other pictures or sounds on top of these if needed.

OBJECTS

The third major component is the set of objects in the Adventure. In Genesis it is possible to have similar objects at different location in the game, so objects have to be specified by location as well as name. Thus each object name may have many Object Status Sets associated with it, as indicated in Fig. 2. Each Status Set specifies a location for the object, but also the object status. Obviously there must be something to indicate whether, for example, the object is being carried or not and this is done by Status markers. You can have up to 48 of these, allocated to whatever purpose you wish. Apart from a "carry' status you might have, for example, another status marker to show whether it is possible to carry the object.

COMMANDS

The structure of Commands is shown in Fig. 4. The first thing to notice is that there are two 'default' Messages associated with each command. These are the Messages which would be shown if the player used (1) an object which is not in the Adventure or (2) an object which is in the Adventure, but cannot be used with this command (eg. 'take floor' would be ridiculous).

Commands may have no objects (eg. 'look' or 'inv'), one object (eg. 'take something') or two objects (eg. 'hit wall with head'). In the last case, the Command would have two names, 'hit' and 'with', as well as two objects. This is illustrated by Fig. 4.

Often in an Adventure, the same command will do different things at different locations and/or for different objects. In TEXTURE these different Actions are catered for by having a different Object Sequence for each case. Thus a Command may have many Object Sequences attached to it.

As shown by Figs. 5 to 7, an Object Sequence format differs slightly for Ø, 1 and 2-object Commands. If there are no objects, the Object Sequence simply defines a location (which may be specified as 'ANY'), and an Action number. Actions will be described later, and are used to do such things as printing messages, changing status markers, moving objects or the player to new locations, etc.

You may be asking what happens if, for instance, you define an Object Sequence which applies to ANY location, and also one for a specific location, both on the same Command. The rule is that the Sequence defined for the specific location takes precedence at that location, so there is no conflict. When you have objects as well as locations in the Object Sequence (see Figs. 6, 7) the same type of rule applies, ie. if a Sequence for a specific object conflicts with a Sequence defeined as applying to ALL objects, the specific sequence takes priority).

Object Sequences for 1-object Commands include an object name, which of course may be specified as 'ANY'. The corresponding Action will only be carried out if the player uses that particular object in the Command. Similarly, for 2-object Commands, the names of two objects must be specified.

ACTIONS

To recap slightly, we have seen how locations have status markers attached to them, how objects have both status markers and locations, and how Actions are linked to Commands through the use of Object Sequences. Obviously, in Actions we shall need to be able to, for example, alter status markers of specific locations or objects, change object locations, move the player location, or print responses to the players' Commands.

It should also be obvious that we need to be able to test such things as whether an object is being carried, whether a location is dark or not, or whether a particular object is at the players location.

We term things like the location of the player, or the status of a location, 'Variables'. Tests to determine the value of a Variable will be termed 'Conditions'. The actual actions of printing messages or altering the value of a Variable will be done by 'Action Operators', or just Operators for short. A full list and explanation of these three elements which make up any Action is given in the reference section. At present it is enough to have a broad idea of what an Action is.

Some Examples

Before entering the heavy stuff in the reference section, we shall take a look at a few examples of Actions used in the example Adventure. Hopefully, an understanding of these will make the Reference Section easier going. Incidentally, a complete list of Actions, Commands, etc. is given in Appendix 1 for the example game. If desired, you can load Texture on Side 1 of the cassette ($RUN"TEXTURE"$$), and then load in the Texture data files from Side 2. Make sure you wind the tape forward to just before the data files, or else you will be trying to load the CLONER program as data! If you load the files correctly, you will be able to call up the Action on the display, and compare it with the explanation below. (Use the 'Define Action' option).

The HELP Command Action

The HELP Command has been linked to Action number 5 in the example game. This was done by setting up an Object Sequence for Help which specified all locations, and Action 5. The actual Action reads:-

CALL<3Ø>:MSG<PLOC;Ø;1>:FINISH:

This is a nice easy one to start with. We shall ignore the first Operator CALL<3Ø>, as this simply calls Action number 3Ø. The second operator, MSG<PLOC;Ø;1> is used to select the Help message for the current player location (PLOC). As mentioned previously, three special messages can be linked to each location, and these are indicated by the numbers Ø, 1 and 2 for the Help, Description and Invalid Command messages. Thus MSG<PLOC;Ø;1> produces the Help message, while MSG<PLOC;1;1> would produce the description.

The other number (1 in this case) is priority of the message. When an action prints several messages, the higher priority ones are printed first, regardless of what order they appear within the Action.

The last Operator is FINISH. All Actions should end with this Operator.

The Take Action

Taking objects is a command which you will find in every Adventure. In the example it has been linked to Action 2 for most objects and locations, though there is a special action for 'take Vid file' at the 'G.I.A. Office' location (Action 4). We shall look at Action 2, which can probably be used as it stands in your own games, provided you use the same status names.

The complete Action is:-

```
CALL<3Ø>:
[OBST(CARRIABLE,OB1)=Ø1 MSG<17,1>:FINISH:
[OBLC(OB1)#PLOC]  MSG<9;1>:FINISH:
[OBST(CARRY,OB1)#Ø]  MSG<8,1>:FINISH:
COPY<1;OBST(CARRY,OB1)>:CALL<39>:FINISH:
```

Ignoring the CALL<3Ø> line again, the next line reads in words:- "If the object referred to by the player is not carriable, print message 17 and finish."

To see how this is decoded, the first thing to realise is that anything in square brackets is a Condition. Thus the object status (OBST) marker name 'CARRIABLE' is tested to see if it is the zero (ie. false). By using the symbol (OB1), we ensure that it is the object referred to by the player that is used. As explained later, there are several forms that the OBST variable can take, for example instead of OB1 you could use a specific object name. However, for the take command, we must use OB1 as it is the object named by the player that must be tested. (For 2-object commands, you can use OB1 or OB2).

If the Condition is true, then what follows it on the same line is carried out. In this case, it is a slightly different form of the MSG Operator met earlier. When PLOC is not included in the brackets then the first number is simply the actual message number to print. Thus, MSG<17;1> simply prints message 17 at priority 1. (This is message "Impossible!").

The next line of the Action tests to see whether the object exists at the player location. Thus the object location (OBLC) of the player-named object (OB1) is tested to be not-equal-to the player location (PLOC). If it is not-equal (ie. the Condition is true), then message 9 is printed indicating that the object is not there.

The following line checks to see whether the object is already being carried. You should by now be able to see how that works. Finally, if none of the three Conditions are true (ie. if the object is carriable, and is at the player location, and is not already being carried), then the COPY Operator is used to change the 'CARRY' status to 1 (ie. 'true'). The CALL<39> simply calls an action to display the new state of the objects, as in the example game carried objects are displayed on the left, and those just lying around on the right.

The Look Action (Action Ø)

This is an interesting one as it uses an object scan loop. The purpose is to print all the objects that are lying around at the current location, so obviously all objects must be checked one by one. This is achieved using the OBLOOP: and ELOOP: Operators.
The actual Action reads:-

```
MSG<6;29>:COPY<Ø;M(1)>:OBLOOP:
[OBLC()=PLOC][OBST(CARRY)=Ø][OBST(LKAB)#Ø]VPRINT<(A)>:SPC:VPRINT<.....
OBJ>:NEWL:COPY<1;M(1)>
ELOOP:
[M(1)=Ø] VPRINT<(NOTHING)>:NEWL:
FINISH:
```

Several new Operators are used in this Action. On the first line, MSG is an old friend, and COPY has been met before. (The M(1) variable is a general purpose one usable for anything. The reason for loading it with zero will become apparent later). However, OBLOOP is a new one. In conjunction with ELOOP further down, it indicates that all lines between the two are repeated for every object at every location.

The second and third lines down are in fact just one line, it's just that the line is too long to fit. Obviously this is the line which will be repeated.

The line starts with three Conditions, enclosed in square brackets. All of these must be true before the Operations following them on the line can be executed. The first Condition checks to see if the current object's location (OBLC) is the same as the player (PLOC). Notice that since the line is repeated for every object we do not specify the object. Thus instead of OBLC(OB1) as in the Take Action, we just have empty brackets OBLC(). The latter form is only valid inside scan loops.

The second condition checks that the object is not being carried. Again, the particular object is not specified, ie. OBST (CARRY) instead of OBST (CARRY, OB1). Finally, the third Condition checks that the object is 'Lookable', using status marker LKAB. There may be some objects which you do not want to show in the Look command, even if they are present, for example, a fixture described in the location description which is not really relevant to the game.

Assuming all the Conditions are true for the current object, then the first VPRINT Operator prints the letter A. In general VPRINT can be used to print any variable including words enclosed in brackets. Next comes the SPC Operator. Using VPRINT removes any spaces, for example VPRINT<(This is)> would print "Thisis", so a space operator is needed to separate words. Next, VPRINT is used again to print the name of the current object. The variable (OBJ) is only valid within loops, and represents the current object. The NEWL Operator prints a newline character, and then the M(1) variable is loaded with the value 1, for reasons explained below.

We have, hopefully, now be-laboured object loops enough. After the ELOOP operator, the Action checks to see if the general purpose variable M(1) is zero. If it is, it means that no objects suitable for listing by LOOK have been found at the player location, so VPRINT is used to print the word 'Nothing'.

Moving Around

There are several main actions associated with the player moving between locations. Firstly, Action 12 is used in all cases, regardless of which way the player moves. Its effect is to mark that the location has been visited, to update the locations of any objects being carried, and to print the description message and put up the picture for that location. You should by now be able to follow the broad outline of the action. We shall confine explanation to the new things that the Action uses:-

```
[PLOC#(OUTEROFFICE)] COPY<Ø;M(2)>:
COPY<1;LCST(VISIT,PLOC)>:OBLOOP:
[OBST(CARRY)#Ø] COPY<PLOC;OBLC()>:
ELOOP:CALL<3Ø>
[M(9)#Ø] CALL<28>:COPY<Ø,M(9)>:MSG<71,2>:
CALL<32>:CALL<39>:PICT<PLOC;Ø;Ø>:MSG<PLOC,1;3Ø>:DO<0>
```

In the first Condition, we see a new feature, the use of a string constant to check whether the player is at a given location, in this case 'OUTEROFFICE'. The spelling of the name must be match exactly the name you gave to the location, or else the Condition will never be true! String Constants are always enclosed in brackets as shown.

On line 2, the LCST Variable is also new, though very similar to the OBST Variable. It refers to the status marker of the specified location (in this case PLOC). Also new is the PICT Operator. This is similar in form to the MSG Operator (see the Reference Section), but displays pictures rather than printing messages. The second number in PICT is the graphics window for the picture rather than the message priority.

The other Action involved in moving about are:-

Action 7 for North
Action 8 for South
Action 9 for East
Action 1Ø for West

These are the Actions which, in combination with the appropriate location status markers, determine the map of the Adventure. You should be able to work out yourself how they are constructed.

Scoring

In the example game, we have not included any scoring. You might like to by modifying the game to include a scoring scheme as one of the first things you do with Genesis. The score Variable SC(Ø) to SC(9) hold the key - in fact the score given when the player enters SCORE is simply the sum of the values in SC(Ø) to SC(9). Since the score is expressed as a percentage, you should make sure that the total does not exceed 1ØØ.

To modify the game, you will have to load the data files into Texture from Side 2, and then modify the appropriate Action to COPY values into the score variables. For example, you might pick 1Ø of the locations, and put values of 1Ø into the 1Ø score variables as each location is visited. For this, Actions 7 to 1Ø would be involved.

TEXTURE OPTIONS - REFERENCE SECTION

This chapter explains all the options in more detail. When you first enter TEXTURE you will be presented with a 9-option Main Menu, which we cover below in order.

1. DEFINE MESSAGES

This option leads to a sub-menu giving options to either create a new message, or modify an existing one. Messages are numbered sequentially beginning at zero, and you will be told the number of any new message you create. It is as well to keep a note of this, including the purpose of the message (e.g. Swots End description etc.). You are limited to 250 different messages, though each can be of any length.

The TEXTURE EDITOR is used to enter and correct messages, and this is a good time to mention the CTRL Q key. This key is always used to exit from the current option or menu, back to the previous menu. For example, after typing in a message under the editor, pressing CTRL Q will return you to the Message menu, after storing the message. Pressing it again would take you back from the Message menu to the Main menu. This is about all there is to Messages, though you will need to use the editor, which is explained next.

TEXTURE EDITOR

The editor provides all the facilities for inserting/adding text, deleting it, and overtyping. Special keys are used for control, as explained below.

The editor uses a screen window 80 characters wide, just below the menu windows. You can type across the end of a line, though this will appear to split words between two lines quite often. Do not worry about this, as when the message is displayed during the game, the computer will automatically throw a new line if the word is too long to fit on the current line.

You can of course, use ENTER to go onto a new line if you want. Remember that you will have to use the Insert key (CTRL I) to create space when adding text to the end of a line on which you have put a new line. The editor prevents you typing across the end of a line terminated by a new line.

The editor is cursor controlled, and you will see the cursor move around as you type in text, or use the arrow keys. The cursor, of course, shows you where you are in the text at all times. The special keys used are:-

CTRL I — This creates blank space at the cursor position, moving all text to the right. You can press it more than once, or hold it down, to create more blank space. It is used, of course, when you must add text to the middle of a message.

CTRL D — This deletes all characters beginning at the cursor position and up to the first non-blank character. It can therefore be used to get rid of any left-over blanks after using the CTRL I key, or to delete a block of characters in the middle of the text. To do this, position the cursor to the start of the block, hold down the SPACE key to overwrite the characters with blanks, move the cursor back to the start of the block, and press CTRL D. It takes much longer to explain than to do!

DEL — The DEL key is used to delete single characters. You can of course hold it down to delete more than one, but for deleting many characters it is usually quicker to use CTRL D as explained.

CTRL Q — This exits from the editor.

Left and Right arrow keys. - These are used to move the cursor around the text.

2. OBJECT STATUS LIST

You should have a good idea of what an Object/Status marker is by this time. This option allows you to name up to 48 markers, but only allows you to add new markers to the list, not delete them. Try to keep the names as short as possible, in order to save memory space during the design.

The odd spelling mistake is not important, as the player never gets to see these names - only the designer. As long as you know what they mean, there is no problem. In fact the list of names is deleted when you finally generate the game to run independently of TEXTURE. The Status markers are still there, but the computer converts the names to numbers to save space. This means there is more room for graphics etc.

When you creat a new Status name, every object in the system automatically sets the corresponding marker set to 'false'. The 'Define Objects' option is used to change the setting if desired. Naturally, not all Status names will be meaningful for all objects. You are not likely to 'carry' a floor, for example. In fact you may decide to use another marker to indicate whether a marker is valid, e.g. you might have a 'carriable' marker as well as a 'carried' marker.

3. LOCATION STATUS LIST

This is similar to the Object Status list, but for locations. Each location will have its marker set 'false' for each new name. Up to 24 markers can be defined.

4. DEFINE LOCATIONS

This leads to an 8-option sub-menu, as follows:-

1. Select Location

The purpose of this is to select the location to work on. You will be asked to enter the location name, which will then be shown on the screen to confirm that the computer has found it.

If you want to create a new location, just press the ENTER key without typing a name. You will then be asked if you want to create the location, and if so for its name. Location names can be more than one word, although it is best to keep them reasonably short otherwise the computer may not have room on the screen for them during the game.

You can rename locations (option 6), so there is no problem about correcting errors, or even totally changing the locations to a new one.

2. Set Status

This option allows you to set all Status markers for the selected location to true or false. These will be the initial states, i.e. the states at the start of a game. They may well change during the game.

3, 4 and 5. Standard Messages

In these options you will be asked for message numbers for the corresponding purpose (Help message etc.). The message number must exist, otherwise TEXTURE will simply ask again. You can get out of trouble by pressing CTRL Q.

These messages are not printed automatically, except for the 'Illegal Command' message. It is up to the designer to incorporate them into Actions for printing - i.e. into the Actions linked to Help and Move commands.

6. Rename

You will just be asked for the new name.

7, 8. Location Pictures

As previously explained, there are up to two picture/sound blocks attached to each location. Of course the actual blocks are created later with the graphics utility. Each block has a number, and these options allow you to define the two block numbers for the selected location. The numbers must be in the range 0 to 249, and it is as well to keep a note of which numbers you use for each location.

5. DEFINE OBJECTS

In this case, a 4-option sub-menu is called up:-

1. Select object

This is to select the object name you wish to work on. You must have selected an object before the other options can be used.

To create a new object, just press ENTER without a name when asked for the object name. You will then be asked if you want to create a new object, and if so for the name of the new object, which should be typed carefully as for the various reasons it is impossible to change it once you press ENTER and the computer accepts it (though you can put in alternative names for the same object - option 4). Prior to pressing ENTER, you can of course use the DEL key to correct mistakes.

3. New Status Set

Status Sets in effect 'create' an object at a particular location, complete with its own set of Status markers. The object created is given the name of the object selected by option 1. You will be asked for the location name, and also for the 'true' or 'false' state of each Status marker.

It is possible to define an object as being at 'ALL' locations, by using a (*) character in place of a location name. For example, a roof or wall might be present everywhere, and avoids the need to specify it separately for each location. However, you may also want to, say, have a special wall at a particular location, as well as the 'general' wall at all locations. This would allow a different Action to be called up at the one special location, for example 'take wall' might be possible at one location but silly at others.

Amazingly, TEXTURE allows this kind of thing with no problem. The rule is that, where there is an object specified at 'ALL' locations and a separate object created at a specific location, the specific one takes precedence at that location. Thus at that location 'take wall' would set the 'carried' Status of the specific object to 'true', rather than that of the object specified as being at 'ALL' locations. If the 'carried' status of this 'general' object did get set, of course, it would mean that you would be carrying the walls at every location! Very strange!

It is important at this point to stress the difference between the initial status and location of objects, and their values during the game. Clearly they will change in response to the players' commands. What we are setting up under these options are the initial states. These are stored separately in TEXTURE so that they can be restored to restart a game without reloading.

2. Modify/Delete Status Set

Although object names cannot be changed once entered. Status Sets can be deleted or modified as required. The computer first has to find the object you want, so you will be asked for its location. If it finds one at the location specified, you will be given the option to delete it, or to modify its location and status.

4. Alternate Names

You can put in here as many alternate names for the object as you like.

DEFINE COMMANDS

Commands are the things which the player uses to play the game, and form the fundamental link between objects, locations, and Actions (described later). Thus in general a Command can be made specific to a location and an object, and be linked to a specific Action. In fact, the same Command word(s) can be attached to many different locations and/or objects, calling up a different Action for each different location and/or object combination. As explained earlier, this is done through the use of 'Object Sequences'.

The Define Command option calls up a 4-option sub-menu as follows:-

1. Select Command

Again this selects the Command to work on, and you must select one before using the other options.

To create a new Command, press ENTER alone as usual. You will then be asked if you want to create a new Command, and if so for the number of objects attached to the Command, and the Command name(s).

As shown in Fig. 4 and previously explained, Commands may have no objects (look, inv etc.), one object (e.g. take something), or 2 objects (e.g. hit something with something else). Command names must be single words - spaces are not allowed within the name.

2. Enter Object Sequence

Each Command type may have any number of 'Object Sequences' which allow the same Command to call up different Actions at different locations and/or for different objects. In fact there must be at least one Object Sequence if the Command is to do anything useful.

For each Object Sequence created, you will have to specify a location (which may be ALL), Ø, 1 or 2 objects (which may be ANY), and an Action number. The Action number, of course, determines which Action to carry out if the Command is used at the specified location with the specified object(s).

The object names must exist, though of course there doesn't necessarily have to be one at the location specified, as one might be moved there during the game. Similarly, the location must exist. CTRL Q can be used to get out of trouble.

As you can specify for example an Object Sequence to apply to ALL locations, and another one for a specific location, both for the same Command, TEXTURE resolves the conflict by giving priority to the one with the specific location (providing the player is at that location). Similarly for objects, if there is a Sequence applying to ANY object as well as one for a specific object.

3. Delete Object Sequence

To delete an Object Sequence, you will have to tell TEXTURE which one to delete by supplying its location, object(s), and Action number.

4. Select Default Messages

Each Command can potentially have a different message when (1) the player uses an undefined object or (2) uses an object for which there is no corresponding Object Sequence. Thus a wide variety of responses can be achieved for these 'default' conditions. The selected messages are printed automatically in the right circumstances - you do not need an Action to print these.

DEFINE ACTIONS

How Actions are linked to Commands has already been described. In general, each Object Sequence has its own Action number, and this calls up an Action which does all the work of printing messages, altering Status markers, etc. This arrangement gives TEXTURE outstanding flexibility to cope with virtually anything the designer wishes to do, though each individual Action can be quite simple in itself. We give some examples later.

When you use this option, you will be asked for an Action number. Actions are numbered sequentially starting from zero, and to create a new one you simply enter the next number in the sequence. It is wise to make note of each Action number and its purpose as you go.

The TEXTURE Editor is used to enter and modify Actions, though you must stick to a few rules when typing them in. The following example illustrates most of them. Remember that Actions are made up of Conditions, Variables, and Operators. Use CTRL 'Q' to exit.

Example 1 - An Action for an 'Inventory' Command

```
MSG<Ø;1>:
OBLOOP:
[OBST(carried)#Ø] VPRINT<(A)>:SPC:VPRINT<OBJ>:NEWL:
ELOOP:
FINISH:
```

The first line means print message (Ø) - assumed to be something like 'You are carrying:' - at priority 1. Message priorities are explained later. They allow messages to be printed in the right order.

Line 2 is an Operation to loop through all objects at all locations. The loop extends down to the ELOOP Instruction. All Operations etc. between OBLOOP and ELOOP will be repeated for all objects. Thus line 3 tests every object to see whether the 'carried' status is true (i.e. not equal (£) to zero), and if so prints some stuff.

VPRINT<(A)>prints the word 'A', followed by a space (SPC). Then VPRINT<(OBJ)>prints the object name, as OBJ is a variable used within loops to get the current object. Finally a new line (NEWL) is printed.

Format Rules

Refer back to Example 1 as you read through the following:-

1.

Every Operator must be terminated by a colon (:)

2.

Conditions are enclosed in square brackets. You may have more than one Condition in a line, but they must all be at the start of the line, e.g.:-

```
[OBST(carried)#0][OBLC()=PLOC] VPRINT <OBJ> :NEWL:
```

would be valid, but not:-

```
[OBST(carried)#Ø][VPRINT<OBJ>  [OBLC()=PLOC] NEWL:
```

If there is more than one, they must all be true to carry out the Operations following on the same line. In this context 'line' means a line terminated by a newline (i.e. ENTER) character. If you simply carry on typing across the edge of the screen, this will still be on the same 'line' even though it appears on the next line of the screen.

3.

Any parameters or Variables associated with an Operator are enclosed in angle brackets E.g. MSG<Ø;1>:

4.

If there is more than one variable inside the angle brackets they are separated by semi-colons (;), e.g. <Ø;1>

5.

Any parameters associated with a Variable are enclosed in round brackets, e.g. OBST(carried). If there is more than one parameter, they must be separated by commas.

Example 1 shows how easy it is to create Actions in TEXTURE. This one is practically all you need for any inventory command. Obviously you will have to learn the Variables and Operators which can be used, but there are not many of these as you might think. The ones that are provided have been subject to a great deal of thought and hopefully we have achieved the minimum set needed to give you total flexibility. Before looking at more examples, we will look at all of them in some detail.

Variables

These are all the things you need to refer to within Actions to design your Adventure, such as Status markers, Object and player locations, etc.

OBST - the Object Status Variable

There are various forms of this Variable, all of which refer to the status marker of a particular object.
Firstly there is the:-
OBST (status, object, location)
form. Obviously, to identify the correct object, TEXTURE needs to know both its name and location. In this form of the Variable these are contained as parameters within the round brackets. 'status' is any name from the Object Status List.

Often you will want to use the player location instead of specifying it explicitly. This is done using the form:-
OBST(status,object)
which simply leaves out the location.

In the third form, both the object name and location are omitted:-
OBST(status)
This form was met in Example 1. It is used within object loops (OBLOOP - ELOOP) to get the status of the current object. In effect it will refer to the status for every object, one by one.

The two remaining forms are:-
OBST(status),OB1) or:-
OBST(status,OB2)
Here the player location is used since it is not specified, and the object name is taken as that specified in the actual Command. For 2-object commands, OB1 represents the first object, and OB2 the second. For example in 'take torch', OB1 would be 'torch' and OB2 undefined. In 'hit wall with hammer', OB1 would be the wall, OB2 the hammer.

LCST - the Location Status Variable

There are just three forms of this:-
LCST(status,location)
refers to the status at a specific location.
LCST(status, PLOC)
indicates the status for the current player location, whilst:-
LCST(status)
is used within location loops. Just as for objects, it is possible to loop through all locations using (LCLOOP - ELOOP). This second form effectively refers to the status of all locations one by one, when used inside a location loop.

OBLC - the Object Location Variable

You will need to be able to change and test the locations of objects in your Actions. Again there are various forms as follows:-

OBLC(object,location)
Just as for the Object Status variable, the desired object must be specified by its name and location. In this form, these are specifically named. Of course, if you can specify its location you might think that OBLC is redundant, as you already know its location. However, this form would be only useful for writing a new location to the object, when the location specified would be its old location.

OBLC(object)
If the location is not specified, the player location replaces it. Here TEXTURE has another clever trick up its sleeve, for if there is no such object at the player location, it effectively chooses the first object it finds in its list of objects at different locations. This means that you can refer to objects which are elsewhere and move them around as if by magic!

OBLC()
This form specifies neither the object nor its location. It is used within Object loops (OBLOOP-ELOOP) to refer to the location of every object, one by one.

OBLC(OB1) or OBLC(OB2)
By now you should have a good idea of what these mean. Instead of a specific object, these refer to the object specified in the players' command. Again OB1 and OB2 indicate either the first or second object. The location is taken as the player location, just as in the OBLC(object) form, and the same trick is employed if there is no such object at the player location.

OB1, OB2 - Player-defined Object Names

These two variables refer to the names of the objects specified by the player in a Command, as previously mentioned. Mainly they are used as parameters in OBLC or OBST, but you could use them for any other purpose. For example you could print them using VPRINT<OB1> etc. Thus instead of responding 'It's not here' for an object not at that location, you could replace the "It's" by the object name.

PLOC - the Player Location Variable

This is simply the player location. You can use it anywhere appropriate, for example to test whether a particular object is present at the player location. If you VPRINT it, the location name will be printed.

OBJ - the Object Loop Name

This variable is the current object name in object loops, and only meaningful within such a loop. If printed, it will give name of the current object.

LOC - the Location Loop Name

This has a similar function, but for location loops.

M(1) to M(50) - the Marker/counter Variables

These variables can be put to any use the designer wishes. They can store numbers in the range —32767 to 32768, and be incremented and decremented. For example, a Marker/counter variable might be used to count how long a torch had been on, or keep track of time.

SC(0) to SC(9) - the Score Variables

These ten variables can hold numbers between 0 and 32767, and are used for scoring. Virtually any scoring system you like can be implemented, involving for example which locations have been visited, or particular commands used. Further details on scoring are given in the section which dissects the free Adventure provided with the package.

Constants

Two types of constants can be used in actions, numbers or text constants. Numbers must be in the range 0 to 32767.
Text constants are normally enclosed in round brackets, e.g.:-
VPRINT<(Strike)>:
would print the word 'Strike'. Spaces are not significant, and might be removed by TEXTURE. In some cases, text constants are converted to the equivalent location or object, where the name matches. For example:-
COPY<(Swots Acre);PLOC>
would move the player to Swots Acre, provided it existed. The space is not significant, in fact:-
COPY<(SwotsAcre);PLOC>:
would do just as well.

Operators

Finally in this section we look at Operators.

The MSG Operator

Printing messages is an obvious necessity, and to do so we must specify a message number, and a priority. The reason for having a priority is that during an Action you may wish to print several messages, but not always in the order they occur within the Action. As the Action proceeds, each message is added to a list of messages to be printed when the Action is complete, together with its priority. Then all the messages are printed, beginning with the highest priority ones. Messages of equal priority are printed in the same order as they are added to the list.

The simplest form of the Operator is:-

MSG<n1,n2>:

where n1 is the message number, n2 the priority (Ø to 49). The message number must exist, otherwise you will get a Syntax error when you exit from editing the Action.

The second form is:-

MSG<PLOC;n1;n2>:

This is used to get one of the three messages which are attached to each location, namely the Help message, the Description message, and the 'Illegal command' message. PLOC indicates that these message numbers are to be taken from the players' location.

In this form, n2 is again the priority, but n1 can only take values of Ø, 1, or 2 corresponding to the three 'attached' messages.

A little thought will show why this second form is needed - if it wasn't there the only way to select, say, a description message when the player moves would be to test individually whether the player was at each location, a very inefficient method.

The VPRINT Operator

Often you will need to print actual variables, such as the names of objects etc. This is done using:-

VPRINT<variable>:

for example:-

VPRINT<OBJ>:

would print an object name. VPRINT has a priority of 24, and all VPRINT operators in an Action are joined together for printing as a single message. You cannot interspace MSG output and VPRINT output.

For example:-

VPRINT<OBJ>:MSG<Ø;24>:VPRINT<LOC>:

would in fact first print both an object name followed by a location name, followed by message (0). The message would not appear between the object and the location names, even though the MSG priority is 24, as all VPRINTS effectively generate a single message.

The Copy Operator

This is used to write a value from one variable to another, e.g.:-

COPY<(Swots Acre);OBLC(OB1)>:

would move the object referred to by the player, OB1, to Swots Acre.

The general form is:-

COPY<from;to>:

The FINISH Operator

This is used to terminate the Action at that point.

The OBLOOP Operator

OBLOOP is used in conjunction with ELOOP to loop through every object. You cannot ever nest loops inside each other, be they Object loops or Location Loops.

The LCLOOP Operator

This loops through all locations in turn. ELOOP is used to indicate the extremity of the loop.

The ELOOP Operator

Used in conjunction with OBLOOP and LCLOOP.

The PICT Operator

This is used to call up Graphics/sound blocks and has two forms similar to the MSG Operator. The simple form:-
PICT<n1;n2>:
will activate picture/sound block n1, in window n2. As for messages, each picture request is added to a list and only displayed after the action is complete.

The second form:-
PICT<PLOC;n1;n2>:
selects one of the two picture/sound blocks attached to each location. In this case, n1 must be 0 or 1 only. (See Fig. 1 and the 'Define Locations' option.

The CANCEL Operator

Sometimes you will want to cancel all the messages requested so far during the Action. This is done with:-
CANCEL:

The PCANCEL Operator

This cancels all picture/sound blocks so far.

The NEWL and SPC Operators

These are used to put in spaces and new-lines to the VPRINT message.

The RESTART Operator

This is used to 'kill' the player, a VERY powerful Operator! After it the player will only have options to restart the game, or resurrect himself.

The INC and DEC Operators

You can only use these with the 50 M-type variables, to increment or decrement them by one. For example:-
INC<M(1)>:
DEC<M(2)>:
etc.

The DO Operator

You can jump from one Action to another by using the DO Operator. For example:-
DO<5>:
would effectively finish the current Action, and start Action number 5. This is very useful. Examples will be given later.

The CALL Operator

This is similar to DO, but the current Action, is not finished, merely suspended. When the Called Action finishes, the original Action will recommence where it left off. Calls may be nested to a maximum depth of 50 - i.e. an Action can call an Action which itself calls another Action . . . etc. to a depth of 50. If the depth exceeds this, any further Calls are ignored. An example would be:-
CALL<6>:MSG<Ø;1>:
Action 6 would be done, then message Ø printed. If the CALL were a DO, of course, message Ø would not be printed as the original Action would terminate at that point.

This completes the description of Operators. If you have read and tried these out, you should by now be fairly proficient in the use of TEXTURE. Any problems and questions which you have will hopefully be cleared up by re-reading the previously explained examples from the free game.

8. The TEST Option

The final option on the main menu is Test. As you might expect, this allows you to test the game so far, though of course you will not see any of the graphics or sound.

In the Test mode, you will first be asked which location you want to start at. Type in the exact name, though spaces will be ignored. If you have loaded the example game files, look at Appendix 1 to find the names of the locations.

You can now play the game just as though you were the player, though you will have to use the exact command, eg. $Take Vid file$$. This ensures that the game design is correct. When the game is finally generated via CLONER, things are more relaxed. The player could then just as well say $ Take the !$?!! Vid file$$, with the same effect.

Listing Commands

As well as the game commands, there are some special commands you can use. You can list locations, objects, and commands using:-

$?C Object name $$ (to list the location and status of the object)
$?L Location name $$ (to list the location status)
$?C Command name $$ (to list the Object Sequences for the command)

For example, try:-

$?O Vid file $$
$?L Outer Office $$
$?C Get $$

with the example game.

Please note that the command letters MUST be capital letters, both for listing and the other commands described next.

Modification Commands

As well as listing commands, you can use commands to create new Objects, etc. without leaving Test mode. All these commands are preceeded by the 'slash' character $/$, as follows:-

$/O$$ to create a new object
$/M$$ to modify an existing object
$/L$$ to modify a location status
$/A$$ to create or modify an action

Due to the way that Texture works, all objects and locations are reset to their initial status before the command(s) are carried out, except for the Action command. If you have followed the appropriate section in the reference chapter, you should have no trouble using these commands. The prompts etc. which come up are identical to those appearing when using the menu options to do the same job. The great advantage, of course, is that you can continue to play the game without exiting from Test mode.

Control Commands

There are three special commands as follows:-

$.P Location name $$
This moves the player to any location. For example, try:-

$.P DAMLON SPACEPORT $$ in the example game.

Note that this does not change the locations of any carried objects! Also be sure to remember the decimal point.

$.R$$

This simply resets everything to the initial state, except for the player.

Finally, there is:-

$ CTRL Q $
command. This exits from the Test mode back into the main menu.

Special Note

Although not strictly options, it is worth mentioning here three keys for colour control of the Texture screen. These won't effect the final game in any way, they are just there so that you can set the colour to your own taste.

CTRL B - Changes the border colour
CTRL I - Changes the ink colour
CTRL P - Changes the paper colour

The DEPICTER Utility

Hopefully, by the time you try out Depicter, you will have a good idea of the structure of Genesis Adventures and how Actions are used to activate picture blocks. Depicter allows both pictures and sounds to be created, and the results stored on cassette for use by CLONER. It is vital to ensure that every picture number activated by any Action actually exists, otherwise your game will not run correctly. For example, if you have an Operator PICT < 23;0> : in your Actions and there are only 2Ø pictures created under Depicter, you will have problems when you CLONE the game.

Up to 25Ø pictures (where 'picture' is used in the general sense of a combined picture/sound block) can be defined, numbered Ø to 249. These are the same numbers that Texture uses in the PICT Operators. Depicter is also used to position the two text windows, and up to 20 graphics windows. (A window is simply a rectangular section of the screen). Pictures can be drawn in any window, though normally you would not attempt to draw into a smaller window. Windows can overlap each other, so you can merge smaller pictures into larger ones, for example.

The Initial Menu. Screen Modes and Colours

Depicter is the second program on Side 1, and you should know by now roughly where it starts. Forward wind to this point, and enter:-

$ RUN "DEPICTER" $$

After loading, you will have a 3-option menu on the screen, allowing you to set Screen Modes, Set Colours, or continue. Pressing keys 1 to 3 selects the desired option.

1. Screen Modes

Genesis allows you to select the screen mode independently for the top and bottom of the screen. The mode changes in the middle of lines 12 or 13 (line 1 is the top line). It is best to set your text and graphics windows to avoid these lines when using different modes.

If you select different modes, you will not see the effect until you actually come to draw a picture. Depicter itself always uses screen mode 1. As you should be aware from the Amstrad User Guide, the screen modes are:-

Mode Ø - 16 colours, 2Ø-column text
Mode 1 - 4 colours, 4Ø-column text
Mode 2 - 2 colours, 8Ø-column text

Ink Colours

Option 2 allows you to set 16 different ink colours from the 27 available. These inks are common to all pictures and text. The standard setting is:-

INK	COLOUR	INK	COLOUR
Ø	BLUE	4	BRIGHT WHITE
1	BRIGHT YELLOW	5	BLACK
2	BRIGHT CYAN	6	BRIGHT BLUE
3	BRIGHT RED	7	BRIGHT MAGENTA

INK	COLOUR	INK	COLOUR
8	CYAN	12	BRIGHT GREEN
9	YELLOW	13	PASTEL GREEN
1Ø	PASTEL BLUE	14	ORANGE
11	PINK	15	SKY BLUE

Only on Mode Ø can all 16 inks be used. If the window is on a Mode 1 part of the screen, only inks Ø to 3 are valid. For Mode 2, only inks Ø and 1. However, all 16 inks are set up, though if you are not using Mode Ø you can simply set inks 4 to 15 to the same colour.

The inks are set up in order, starting with ink Ø. Press the SPACE key until the colour you want is on the border. Then press ENTER to set the current ink and progress to the next one.

Continue

Option 3 simply takes you on to the first of the Picture/Sound creation menus. To get back to this initial menu at any time, simply press CTRL Q.

The General Options Menu

This menu allows you to:-

1. Create a Pict/Sound block
2. Set Screen Windows
3. Set Sound Envelopes
4. Save/Load

Option 1

This takes you into the Create Picture menu, described later.

Option 2

This brings up options to position the text or graphics windows. The two possible text windows are labelled 0 and 1. Window 0 is used to display the Location name, and would normally occupy just a single line of the display. The location names will be automatically centred in the window when printed. Window 1 is used for the main text of the Adventure. The procedure for setting the windows is identical, whether it is a graphics or text window. You will be asked first for the window number, then for values for the left and right edges and the top and bottom edges, all of which are specified as character positions. Remember that character line 1 is the top line. In all screen modes, there are always 25 lines from top to bottom. However, the screen mode does affect the number of columns left to right, (20 for Mode 0, etc.). For example, entering $10$$ and $11$$ for the left and right edges would place the window in the centre of the screen in Mode Ø, but in the left quarter for Mode 1. Where a dual-mode screen has been asked for, the mode for any window is assumed to be that of the top line of the window. In other words, if the top line is less than 12, the mode is that for the top of the screen. Normally, on a dual-mode screen, none of your windows would straddle line 12.

Option 3

We do not propose to explain in detail what the sound envelopes are. The Amstrad User Guide covers this fairly well, and there have been many articles on the subject in the computing press. You are expected to have an understanding of what volume and tone envelopes are all about. This option brings up a sub-menu, allowing you to create Volume or Tone envelopes, and test the sound. The Exit option simply returns to the previous menu.

In fact Depicter creates two initial envelopes for you automatically - one Volume and one Tone - with a piano-like sound. Thus you can create music without bothering to set up any envelopes. However, for special effects, it is well worth learning how to do this.

Up to 15 different envelopes of each sort can be created, though the fewer you use, the less memory is used.

1. Volume Envelopes

You will first be asked for the envelope number, and be told the maximum number you can use. For example, the first time you enter the option there will be envelope number zero in the system, the one created by Depicter itself. You can select that one or type in $1$$ instead. In the latter case, a new envelope is created, initially with the same shape as envelope number zero.

Let us assume that you have chosen number 1. On the screen now should be a graph of the envelope, plus another menu to the right, and the legends SCALE, ENV. NO., and REPEAT. The REPEAT option is only used for tone envelopes so that it can be ignored at the moment.

The graph scale vertically is —15 to +15, with zero in the middle. Horizontally, the graph represents time. Try pressing keys 2 and 3 to see how the scale changes. The scale factor is given in thousands of a second, so 16ØØ means a scale of 1.6 seconds.

Next, let us try setting up a simple envelope which increases to 15 in 15 steps of 2 hundredths of a second, stays level for 3Ø hundredths, then decays to zero in 15 steps of 7 hundredths. First press 1, and you will be asked for the segment number. The envelope is always made up of 5 segments, each of which is a 'ramp' up or down (or level). Enter $0$$ to select the first segment, then type in:-

$15$$ (To number of steps)
$1$$ (To step size)
$2$$ (To time step. This is hundredths of a second)

Similarly, enter the following for the other 4 segments:-

Segment 1 $1$$,$Ø$$,$1Ø$$
Segment 2 $1$$,$Ø$$,$1Ø$$
Segment 3 $1$$,$Ø$$,$1Ø$$
Segment 4 $15$$ $—1$$,$7$$

Make sure you understand how this creates the shape in the graph.

We pass on to the NOISE option, key **4**. This allows noise in various band widths to be added to the sound, for special effects such as machine noises, footsteps, etc. Choosing a zero noise factor removes the noise. It is impossible to describe the effects of adding noise, you will just have to experiment, using the test sound option.

The Exit option takes you back to the Set Envelopes menu, where we now look briefly at tone or frequency envelopes.

2. Tone Envelopes

Setting these is virtually identical to setting up volume envelopes, as you will see from the display produced. However, whereas volume envelopes should not normally go less than zero or greater than 15, the tone envelopes are not restricted in this way. Also, the Repeat option is valid, allowing the envelope to be repeated during the sound if it is shorter in duration. On the other hand, the Noise option is not relevant.

Again it is impossible to describe the effect using tone envelopes. Experimenting is great fun, and you may discover just the sound you need to make that graveyard scene truly terrifying!

3. Test Sound Option

On selecting this, you will be asked for the volume and frequency envelope numbers. Then, a picture of a keyboard will appear, one we shall meet again when adding sounds to pictures. The letters over the keys show which Amstrad keys to press to get notes. As well as the letter keys, keys Ø to 7 select the octave (key Ø is the lowest octave).

Of course, you can only play one note at a time, though when adding sounds to pictures you will be able to have three synchronised tracks for playing harmonies.

The test sound option is provided so that you can experiment with envelope shapes before using them in earnest.

The Create Picture Menu

This is a four-option menu as follows:-

1. Select picture number
2. Draw picture
3. Create sound
4. Exit

These options are used to define the sound and graphics for the selected picture, so option 1 must be used first to select which picture to work on. You will be asked for the picture number, which can be an existing one (pictures are numbered from zero upwards) or the next unused picture number if you want to create a new picture.

Option 2

Selecting this takes you into the draw graphics section. You will first be asked which graphics window you wish to use. Of course, this is just for drawing the picture - you can draw the completed picture into any window of the same size as the one you will eventually use.

Once you have selected the window, you will be in draw mode, with the selected window outlined by a frame. In the centre of the window will be a pixel cursor surrounded by four dots. You can change this to a full cross by pressing CTRL C, and back again. The cursor colour can be changed by hitting the SPACE key, so that you can always make it visible no matter what background it is displayed against.

When you use any of the draw commands below, the ink colour will be taken from the border. The border colour is changed by pressing the P key.

Naturally, since the pictures drawn have to be packed into very few memory locations, you cannot expect to be able to draw highly detailed pictures at every location in the Adventure. You can use up to 25Ø user-defined graphics characters to build detailed pictures for a few locations, or use the simpler graphics produced by using linedrawing, infilling, etc. to illustrate every location, or even objects etc. We shall now look at all the command keys in detail. Remember that the cursor is moved using the arrow keys (use SHIFT for faster motion).

1. Drawing Lines and Curves. Keys L, N, E, and C

After setting the border to the colour required, place the cursor at the start of the line and press key L. This causes a 'rubber band' to be anchored at this position. Then move to the end of the line and Nail the band by pressing N. You can continue to 'Nail' the band like this by moving to a new spot and pressing N again. When finished with line drawing, press E.

After this you would have to press L again for another sequence of lines.

Curves are just as easy. Instead of pressing N, press C key. You will then be asked for the angle you wish the curve to turn through. After entering this, watch carefully and you will see a small spot move around the curve, possible leaving a trail of dots. This is so you can judge the curve, and change it if need be.

2. Boxes Keys R, F and E

Boxes may be drawn by pointing the cursor on one corner and pressing R. Then move to the opposite corner and press E for an outline or F for a filled box.

3. The Infill Command Key I

Infill is used in areas with colour. In Depicter, a simplified infill is used for maximum speed, but one which can escape beyond the edges of concave shapes in some circumstances. This is no great problem, as you can always delete the infill command, and split the shape into two or more segments on which infill works correctly. It is always best to start the infill in the narrowest possible place on the shape. This is faster and should solve most 'escape' problems.

4. User Graphics Keys G and H

Key H is used to set up shapes for up to 256 characters of 8 x 8 pixels each. Hopefully, the screen produced on pressing H is self-explanatory. The graphics are numbered from Ø upwards, and you can skip through them by using the angle-bracket keys < and >. The arrow keys are used to move the large block cursor around the enlarged design area, to design the graphic. When complete, it can be transferred into the current graphic number by pressing E. To return to drawing pictures, press ENTER. (See the D key below for how to re-draw the picture).

Once you have set up some graphics, they can be placed in the picture by pressing G. This displays graphic Ø in the top left corner. You can move this around using the arrow keys, and cycle through the shapes using the angle bracket keys, < and >. When ready, press Enter to fix the graphic in place.

5. The Clear Window Command Key 7

This command puts a 'Clear Window' into the picture. You might use it to clear any previous picture before drawing the new one. Two pictures drawn into the same window during the game will be merged, unless the second has a 'Clear Window' as its first draw command.

We look at some control commands. These do not actually put commands into the picture, they are simply for the users' benefit.

6. Deleting The Last Draw Command Key DEL

This key deletes the last drawing command, so can be used to correct mistakes. Do not hold it down, as it auto-repeats. The picture is not affected directly, to see the effect you will have to clear the window and re-draw the picture.

7. Clearing The Window Key CLR

This key simply clears the window. Unlike the Z key, it does not put a draw command into the picture.

8. Re-drawing Key D

This re-draws the whole picture.

9. Finish Key CTRL Q

Pressing $CTRL Q$ exits back to the picture creation menu.

Option 3 - Create Sound

This leads to a further sub-menu with the following options:-

1. Enter Track
2. Enter Rhythm
3. Play Tracks
4. Exit

Briefly, Option 1 is used to enter the sequence of notes, Option 2 to set up the rhythm for the note, and Option 3 to play the results.

Option 1

You will first be asked for the track number, followed by the envelope number. The screen that follows you will have seen before, in the Test Sound options. The piano-keyboard shows which letter keys to press to play notes, whilst number keys Ø to 7 are used to select the octave.

Be a little cautious about pressing keys to play notes here, as each one you press may insert another entry into the list of notes. The option is used to simply enter the sequence of notes. Setting the rhythm is left to Option 2. You can edit the sequence of notes by ear, using the left and right arrow keys to play backwards and forwards through the sequence. Press the delete key to delete the note that was played last. Similarly, you can insert a note at the current position. Try these out to get the hang of it. Imagine that you are splicing bits of tape into or out of an audio tape, and you should get the idea.

If you are not positioned at the last note, any notes you play will simply re-write what was there before, so you can insert space for a note, and then put in the correct note. Of course, if you are at the end, any new notes played will be added to the sequence.

Option 2

Using this, you can set the rhythm for any track, using another track as a backing track, or a standard beat. When asked for the keyboard speed (actually the speed of repeats if you hold the beat key down), a typical figure for normal music would be 15. Only if you want to play a sequence of notes very rapidly should you use smaller values, otherwise you will not be able to release the key fast enough to prevent several notes from playing each key press.

The rhythm is set up simply by pressing a single key (B). As soon as you press the key first time, the first note sounds from the track (the backing track, of course, plays continuously). On the second press, the first note finishes and the next begins. Thus the length of the first note depends on the second key press. For this reason, you must press the key once more after the last note sounds, to finish setting the rhythm. Otherwise, the last note will last forever! You will be returned to the Sound Menu automatically after the final key press.

Option 3

This option has a two-fold purpose. First it enables you to listen to the complete sound, with all tracks playing together if you have used more than one. Secondly, it is used to set the number of times the sound is to repeat itself in the game. A zero value means the sound will repeat until terminated by some other sound starting up.

This completes the description of Depicter. Do not be afraid to experiment with the options, and do not worry if you do not understand all the features immediately. You do not need to be a complete Genesis expert to start building your games. Use the features you do understand and you will soon begin to understand the reasons for having the more complex features. After that, you will wonder why it wasn't obvious to start with!

Appendix 1

The Example Game

1. Action List

ACTION	PURPOSE	ACTION	PURPOSE
Ø	LOOK	21	DEATH
1	INV	22	EAT
2	TAKE/GET	23	KISS MILLY
3	DROP	24	HIM/HER
4	SPECIAL VIDFILE TAKE	25	EAT HIM
5	HELP	26	START-UP
6	UNUSED	27	ON THE LIPS
7	GO NORTH	28	GENERAL INJURY
8	GO SOUTH	29	SPECIAL EAST
9	GO EAST	3Ø	UPDATE M(6)
1Ø	GO WEST	31	UNUSED
11	GET KEY (OUTER OFFICE)	32	DISPLAY DIRECTIONS
12	GENERAL MOVE	33	INSERT KEY
13	KILL CHIEF	34	LIBRARY FILE
14	I AM	35	TWIST HELIX
15	I DON'T	36	ASSOCIATED WITH 34
16	YES	37	ASSOCIATED WITH 34
17	NO	38	KILL SHICKLGRUBER
18	GENERAL KILL	39	DISPLAY CARRIED OBJECTS
19	GENERAL HIT/KICK	4Ø	DISPLAY PRESENT OBJECTS
2Ø	SWEARING!!	41	DESCRIBE

2. Messages

There are 72 messages, examine these using Texture.

3. Commands

(This is not a complete list)

GO, TAKE, GET, HAVE, RUN, WALK, EAT, KISS, HUG, KILL, I, ANSWER, SAY, UP, DOWN, LOOK, RUN, INV, DESCRIBE, DROP, HELP, YES, NO, NORTH, SOUTH, EAST, WEST.

4. Objects

(Not a complete list)

WALL, DOOR, CHIEF, NORTH, SOUTH, EAST, WEST, HIM, HER, TASK, TRANSMAT, ROOF, CEILING, HELL, UP, DOWN, LEG, DINNER, MIRACLE, SECRETARY, MILLY, SOO, BOIL, CHEER, BUZZER, OFFICE, SMELLYVISION, CONTEST, JURY, PHONE, AM, LIPS, OUT, VIDFILE, FILE, TAPE, KEY, HELIX, GUN, F1 to F16.

5. Object Status Markers

CARRY, CARRIABLE, LKAB

6. Location Status Markers

NO (NORTH), SO (SOUTH), EA (EAST), WE (WEST), VISIT (VISITED)

7. Locations

G.I.A. OFFICE, OUTER OFFICE, DRAUGHTY CORRIDOR, OUTSIDE THE LIBRARY, THE LIBRARY, VIDFILE READER, OUTSIDE THE G.I.A., YOUR OFFICE, THE TRAVELSTAT, VERSAT SPACEPORT, DAMLON SPACEPORT, JUNKSHOP, HARMIES HI BAR.

Appendix 2

Hints for Advanced Users

1. Saving memory

In most adventures, it is messages which are likely to use the lion's share of the memory. Large savings can be made by using sub-messages for phrases or even long words that are used more than once. Remember that including an extra MSG Operator in an Action only takes up three bytes, so separate messages for even quite short words will save space if they are used frequently. There are limits, however. Firstly, of course, you can only have a maximum of 250 messages, but also Genesis can only queue up to 10 messages prior to printing them. Any more, and the last ones will be ignored.

You can also save space in Actions, by copying long location names to one of the M-type general variables. Since locations (and objects for that matter) are given numbers as well as names, an operation such as:-

COPY<(OUTER OFFICE):M(1)>:

would load M(1) with the location number. Since the PLOC variable is also a number, copying an M-type variable to PLOC is also quite possible. Examples of this technique can be seen in actions 9 and 10 of the example game.

By a similar technique as that used for messages, space can also be saved by constructing larger pictures from several smaller ones. Careful choice of graphics windows is needed, but large savings can be made.

2. Adding Loading Screens

The CLONED game deliberately does nothing to the screen whilst loading. Thus it is quite easy to add your own leader program to create a loading screen.

The program should end with a RUN"!" instruction.

Technical Queries

All technical queries should enclose a STAMPED, Self Addressed Envelope, and should be sent to:-

CAMEL MICROS GENESIS QUERIES

WELLPARK,
WILLEYS AVENUE,
EXETER
EX2 8BE
Tel: (0392) 211892